images & réseaux

Rennes Atalante TECHNOPOLE

Journée Thématique Logiciel Embarqué

Ingénierie Des Modèles: MOPCOM-I

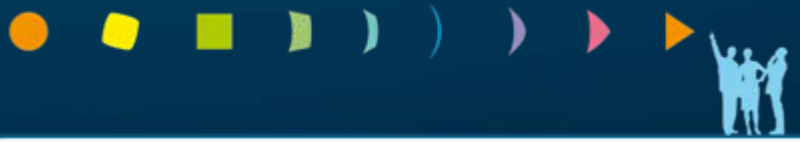
Benoît Baudry (INRIA), Jean-Pierre Mével (THALES)

INRIA

THALES

images & réseaux

Introduction à l'IDM



Once upon a time…
software development looked simple

- From the object as the *only* one concept
  - As e.g. in Smalltalk
- To a multitude of concepts

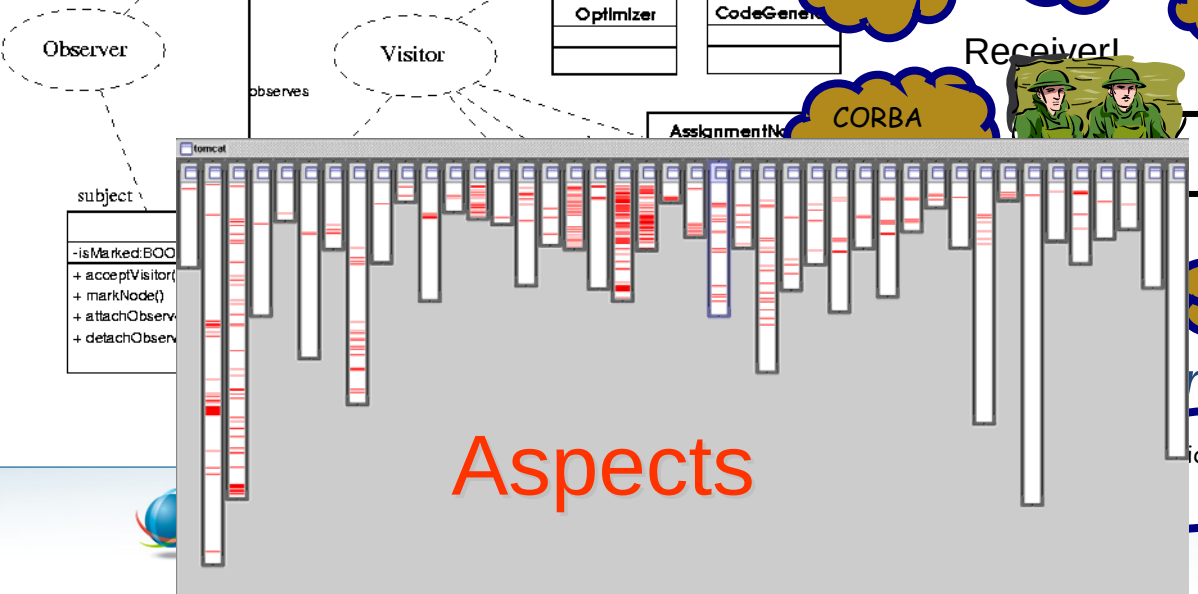
Collaborations

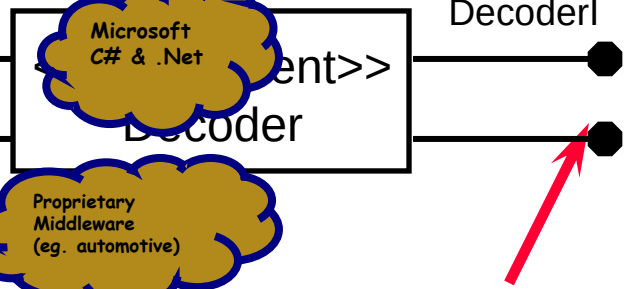
Middleware (middle war)

It's difficult -- in fact, next to impossible – for a large enterprise to standardize on a single middleware platform. (R. Soley)

Components

Design patterns





Provided Port

Aspects

Why modeling: master complexity

- Modeling, in the broadest sense, is the *cost-effective use of something in place of something else for some cognitive purpose*. It allows us to use something that is *simpler*, *safer* or *cheaper* than reality instead of reality for some purpose.

- A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality.

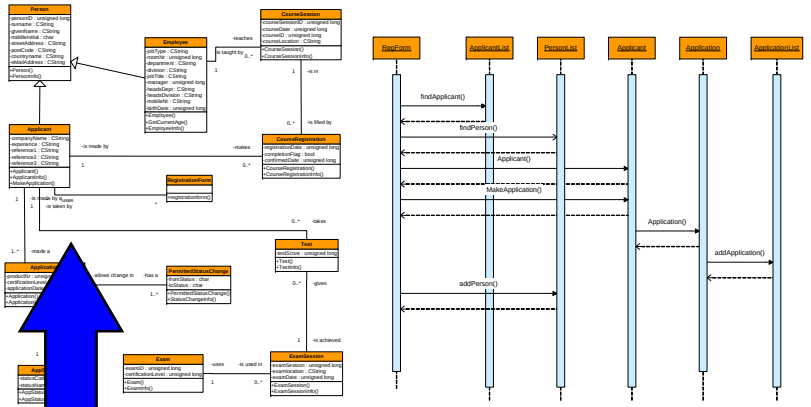
Jeff Rothenberg.

Modeling in Science & Engineering

- A Model is a *simplified* representation of an *aspect* of the World for a specific *purpose*

Specificity of Engineering: Model something not yet existing (in order to build it)

M_1 (modeling space)



Is represented by

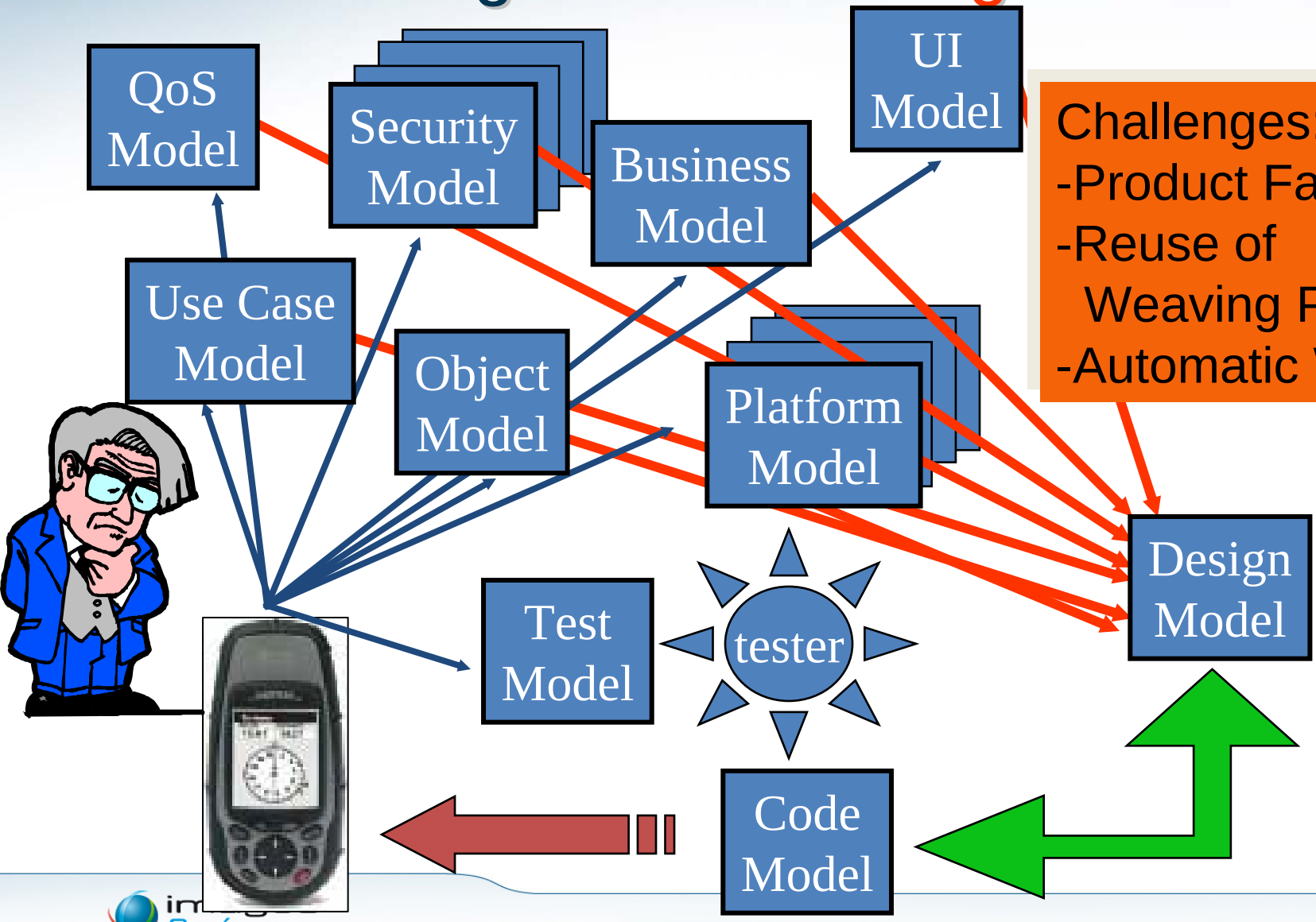
M_0 (the world)

Model and Reality in Software

- Sun Tse: *Do not take the map for the reality*
- Magritte

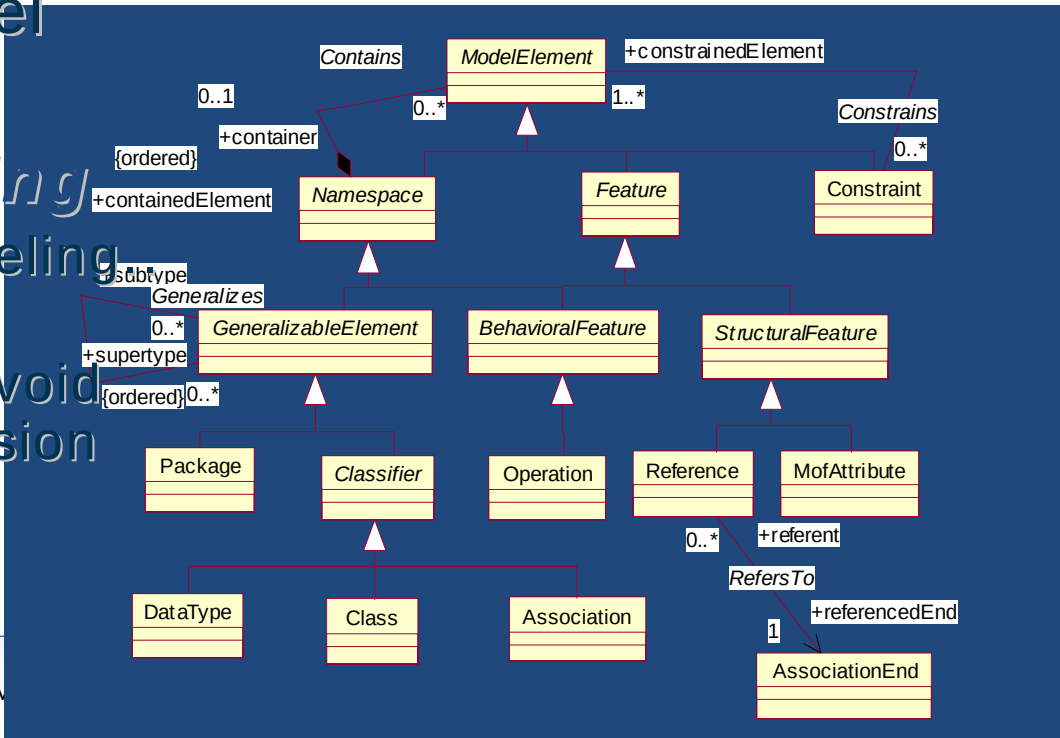
- Software Models: from contemplative to productive

Modeling and Weaving

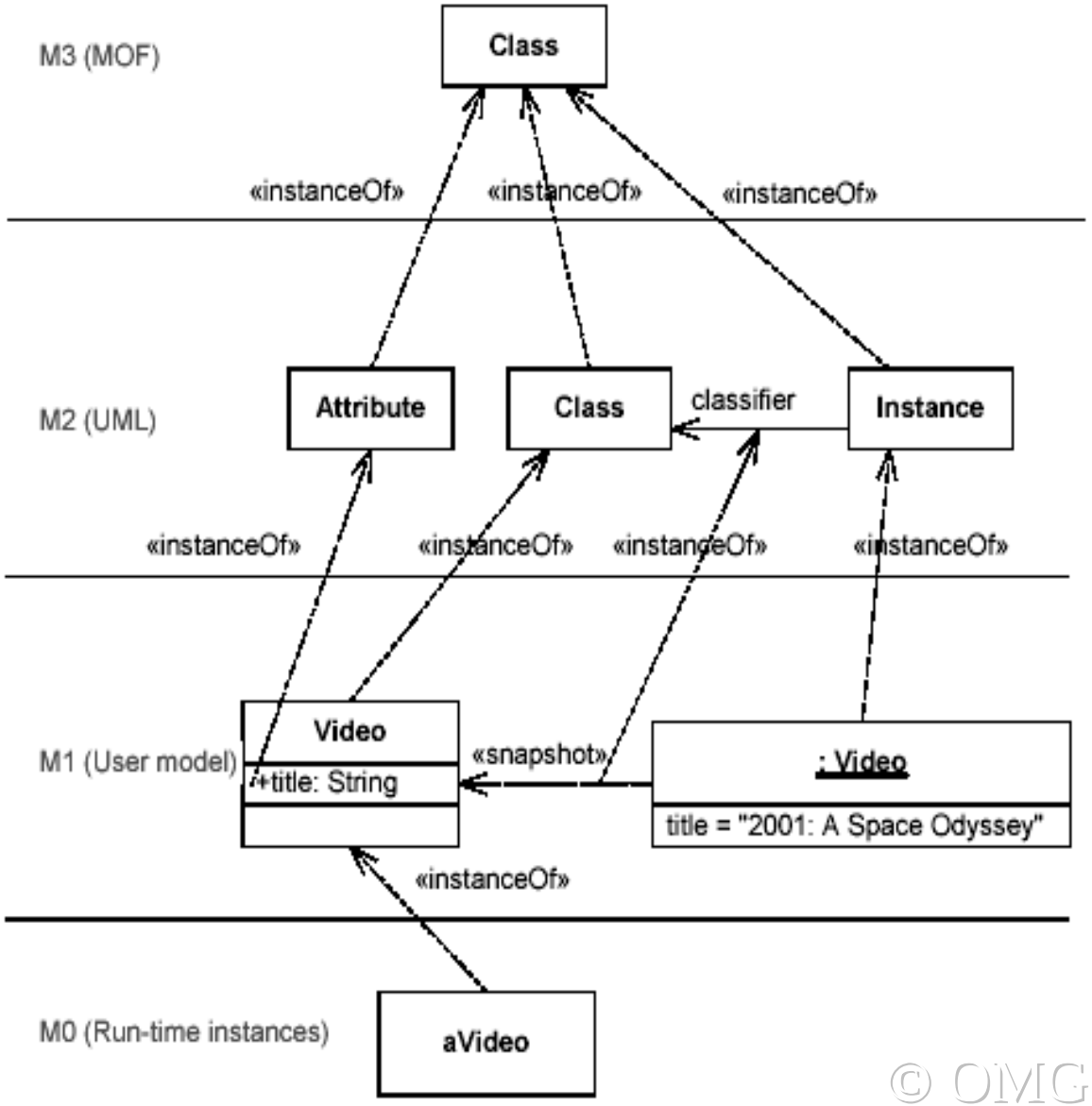


Assigning Meaning to Models

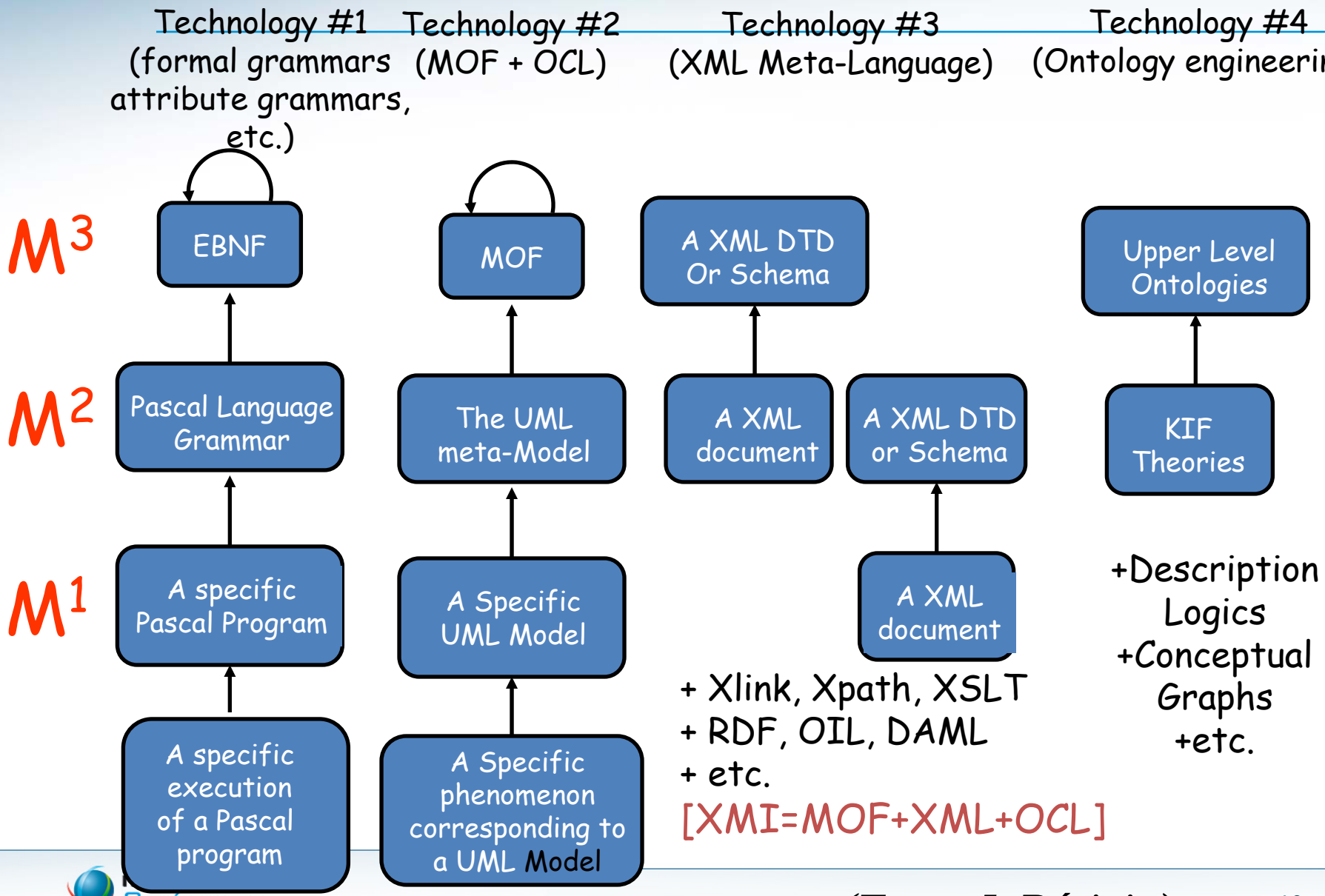
- If a model *is no longer* just
  - fancy pictures to decorate your room
  - a graphical syntax for C++/Java/C#/Eiffel...
- Then tools must be able to manipulate models
  - Let's make a model of what a model is!
  - => ***meta-modeling***
    - & meta-meta-modeling
    - Use Meta-Object Facility (MOF) to avoid infinite Meta-recursion



The 4 layers in practice



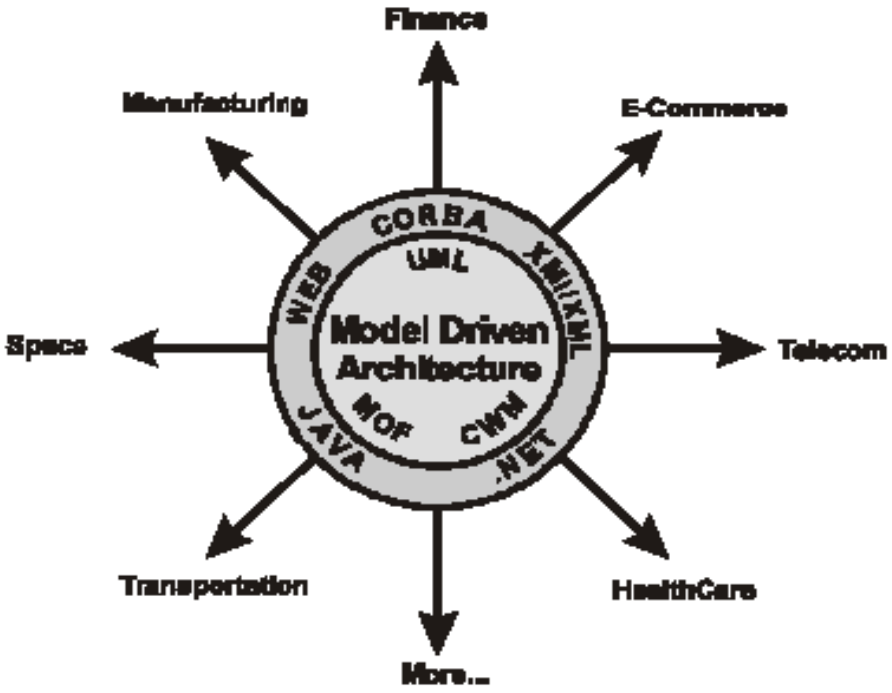
Comparing Abstract Syntax Systems



(From J. Bézivin)

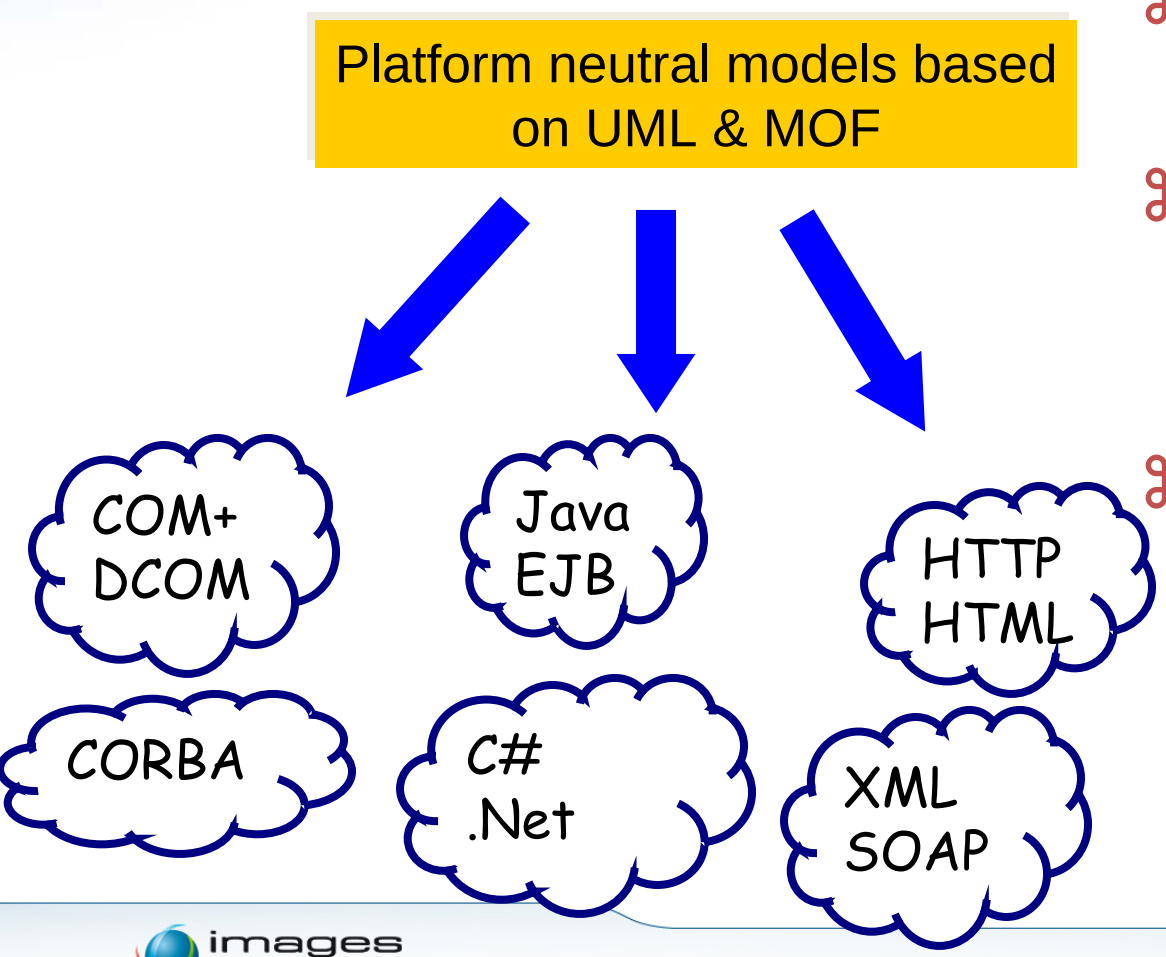
MDA: the OMG vision

"OMG is in the ideal position to provide the model-based standards that are necessary to extend integration beyond the middleware approach... Now is the time to put this plan into effect. Now is the time for the Model Driven Architecture."



Richard Soley & OMG staff,
MDA Whitepaper Draft 3.2
November 27, 2000

Mappings to multiple and evolving platforms



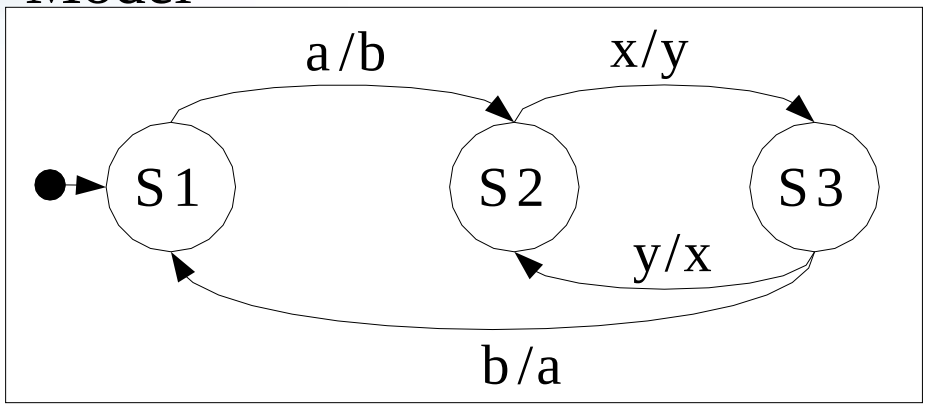
- MOF & UML as the core
- Organization assets expressed as models
- Model transformations to map to technology specific platforms

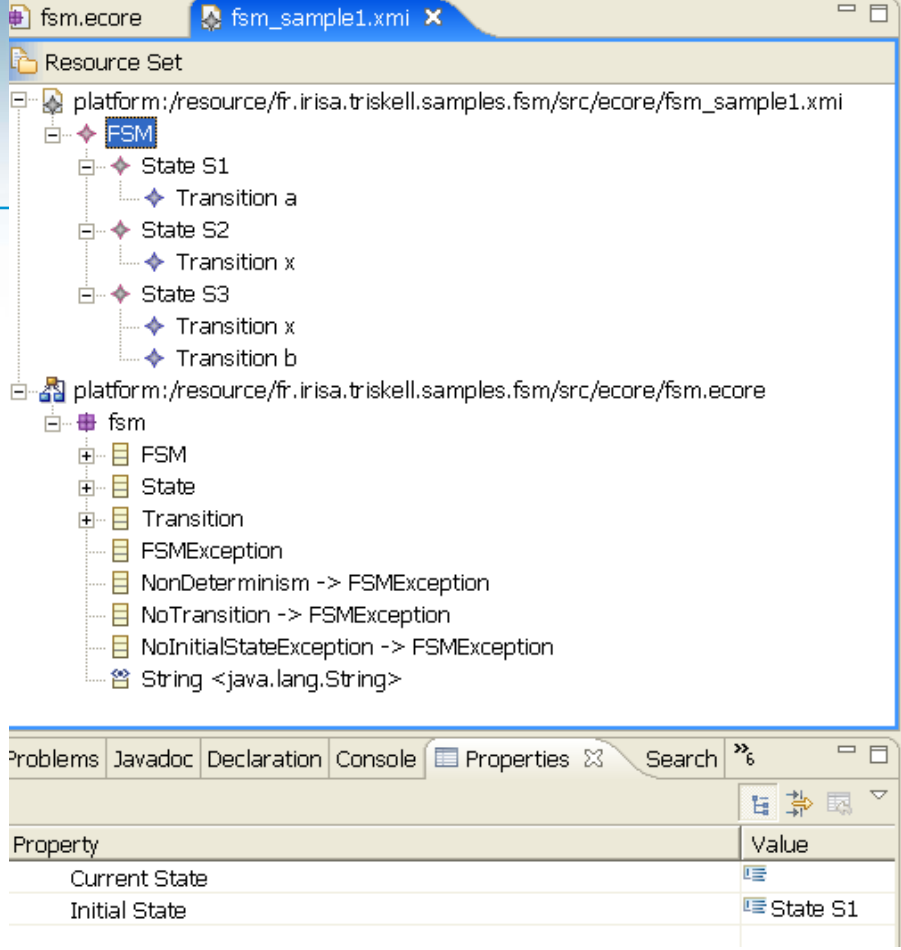
Meta-Models as Shared Knowledge

- Definition of an Abstract Syntax in E-MOF
  - Repository of models with EMF
  - Reflexive Editor in Eclipse
  - JMI for accessing models from Java
  - XML serialization for model exchanges
- But no integrated way to
  - Check constraints, Well Formedness Rules, Static Semantics
  - Provide dynamic semantics and direct interpretation
  - Model transformation, weaving and code generation

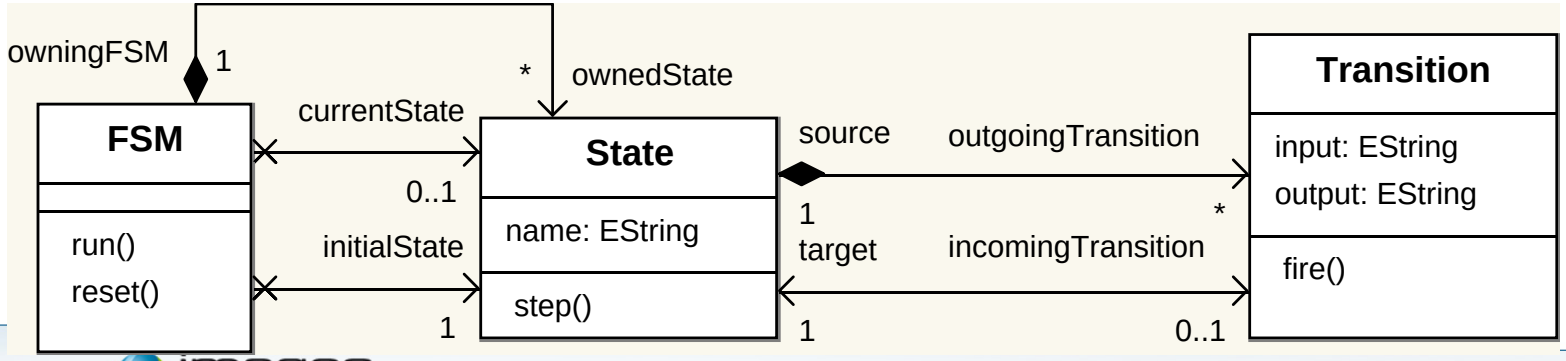
Example with StateMachines

Model





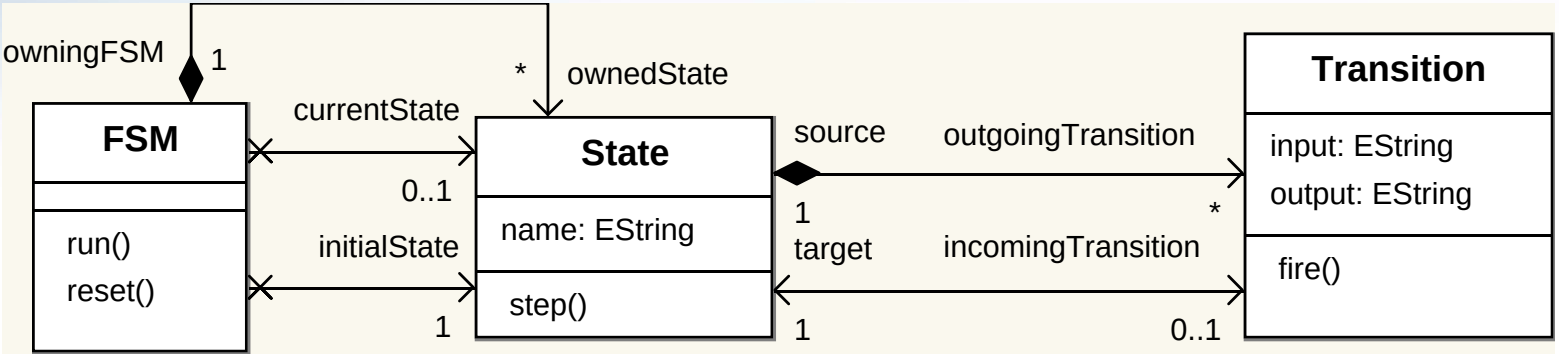
Meta-Model



Kermeta Rationale

- Model, meta-model, meta-metamodel, DSLs…
  - Meta-bla-bla too complex for the normal engineer
- On the other hand, engineers are familiars with
  - OO programming languages (Java,C#,C++,..)
  - UML (at least class diagram)
  - May have heard of *Design-by-Contract*
- Kermeta leverages this familiarity to make Meta-modeling easy for the masses

Breathing life into Meta-Models




```
// MyKermetaProgram.kmt
// An E-MOF metamodel is an OO program that does nothing
    require "StateMachine.ecore"  // to import it in Kermeta
// Kermeta lets you weave in aspects
    // Contracts (OCL WFR)
    require “StaticSemantics.ocl”
    // Method bodies (Dynamic seman
    require “DynamicSemantics.kmt”
    // Transformations
```

```
Context FSM
inv: ownedState->forAll(s1,s2|
s1.name=s2.name implies s1=s2)
```

```
aspect class FSM {
  operation reset() : Void {
    currentState := initialState
```

```
class Minimizer {
   operation minimize (source: FSM):FSM {…}
}
```

Model Driven Engineering : Summary

- Modeling to master complexity
  - Multi-dimensional and aspect oriented by definition
- Models: from contemplative to productive
  - Integrated Meta-modeling tools such as Kermeta
- Model Driven Engineering
  - Weaving aspects into a design model
    - E.g. Platform Specificities
  - Model Driven Architecture (PIM / PSM): just a special case of Aspect Oriented Design
- Related: Generative Prog, Software Factories

Conclusion

- Kermeta is an open-source initiative
  - Started January 2005
  - More than 10 active developer
- Feel free to use
  - Start with a meta-model in EM
    - Get XML an reflective editor fo
  - Weave in static semantics in O
  - Weave in an interpreter,
    - connect to a simulation platform
  - Weave in one or more compilers
  - Finaly care for concrete syntax issues
- Feel free to contribute!
  - www.kermeta.org